

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет имени
первого Президента России Б. Н. Ельцина»

**ПРИМЕНЕНИЕ ВЕЙВЛЕТ МЕТОДОВ
ДЛЯ ДЕКОМПОЗИЦИИ ВРЕМЕННЫХ РЯДОВ**

**Методические указания к выполнению
практического задания № 5**

Екатеринбург

2024

Содержание

Введение.....	3
1. Задание на лабораторную работу	3
2. Требования к оформлению отчета.....	13

Введение

Целью данной лабораторной работы является знакомство и изучение средств *Python* для работы с вейвлетами. Студенты приобретут навыки по использованию этого инструмента для анализа и декомпозиции временных рядов, а также получения оценки их частотно-временных характеристик.

1. Задание на лабораторную работу

Результатом выполнения лабораторной работы является оформленный отчет в виде *Jupyter*-тетради, в котором должны быть представлены и отражены все нижеперечисленные пункты:

- 1) Сначала импортируйте в свой код нужные библиотеки, функции и т.д.

```
import numpy as np  
import numpy.random as rand  
import matplotlib.pyplot as plt  
import h5py  
import pywt
```

- 2) Создадим зашумленный временной ряд с 2 периодиками:

```
t = np.linspace(0, 1, 1024)  
f1 = 10  
f2 = 50  
F=np.sin(2*np.pi*f1*t)+np.sin(2*np.pi*f2*t)+0.2*rand.randn(len(t))  
plt.figure(figsize = (10, 5))  
plt.plot(t, F, 'k')  
plt.plot(t, np.sin(2*np.pi*f1*t), 'b')  
plt.plot(t, np.sin(2*np.pi*f2*t), 'r')  
plt.show()
```

- 3) Наиболее широкой библиотекой для работы с вейвлетами в *Python* является *PyWavelets*. При работе с вейвлетами надо понимать, что это очень гибкий инструмент со множеством параметров. Во-первых, самым главным параметр вейвлет-разложения является сам материнский (базисный) вейвлет. Множество доступных в библиотеке *PyWavelets* семейств вейвлетов и их модификаций доступно на сайте: <http://wavelets.pybytes.com>. Внимательно изучите эти семейства, так как выбор базового вейвлета существенно влияет на конечный результат декомпозиции по принципу самоподобия выделенных компонент и выбранного вейвлета.

Для начала используем вейвлет Мейера:

```
wvlt = pywt.Wavelet('dmey')
```

- 4) Во-вторых, декомпозиция с помощью дискретных вейвлетов происходит до определенного уровня (**level**), ограниченного размером доступных данных. В нашем случае возможно разложение до 4 уровня:

```
pywt.dwt_max_level(len(F), wvlt) # будет выведено число 4
```

- 5) В-третьих, по краям временного интервала вейвлет может по-разному трактовать конструируемые точки для экстраполяции (**mode**): простое дополнение нулями, константами, симметрично/асимметрично, периодически и т.д. В зависимости от вида исходных данных, лучше подходит тот или иной режим.

- 6) Ну и наконец, декомпозиция всегда происходит в виде комбинации коэффициентов **Аппроксимации** (*cA*) плюс **Детали** (*cD*). Всегда есть одна аппроксимация, а число деталей равно уровню декомпозиции. Меняя выбор группировки коэффициентов аппроксимации и деталей, будут меняться восстановленные компоненты и соответствующая декомпозиция ряда.

- 7) Разобьем наш исходный ряд на компоненты с помощью вейвлета Мейера, в режиме периодизации, до 4 уровня декомпозиции:

```
cA4, cD4, cD3, cD2, cD1 = pywt.wavedec(F, wvlt, mode='periodization', level=4)
```

- 8) Обратите внимание, как выглядят выходные результаты: одна аппроксимация **cA4** с номером уровня и 4 детали с убывающими номерами от 4 уровня **cD4** до 1 уровня **cD1**. Если бы мы декомпозировали ряд на 3 уровня (**level=3**), то тогда выходные значения следовало бы записать как: **cA3, cD3, cD2, cD1**. Можно на выходе функции использовать и одну переменную, но ее все равно придется разбивать на отдельные элементы. Также следует помнить, что в результате декомпозиции получаются не новые временные ряды, а только вейвлет-коэффициенты декомпозиции.

- 9) Восстановим две периодики исходного модельного ряда:

```
Fre = pywt.waverec((cA4, None, None, None, None), wvlt, mode='periodization')  
Fre2 = pywt.waverec((None, cD4, None, None, None), wvlt, mode='periodization')  
plt.figure(figsize = (10, 5))  
plt.plot(t, F, 'k')  
plt.plot(t, Fre, 'b') # это будет первая периодика  
plt.plot(t, Fre2, 'r') # это будет вторая периодика  
plt.show()
```

- 10) Снова обратите внимание на форму записи восстанавливаемых компонент. Те коэффициенты, которые мы не хотим использовать для реконструкции компонент (лишние детали и т.д.), мы заменяем на **None**. Меняя комбинации используемых и неиспользуемых аппроксимаций и деталей, мы будем получать разные восстановленные компоненты.

- 11) Проведите аналогичную декомпозицию для 3 уровня (**level = 3**). Вейвлет не меняйте. **Сравните полученные результаты.**

- 12) Повторите все проделанные шаги по декомпозиции ряда и восстановлению его компонент для **своего варианта** базисного вейвлета. Наименование вейвлета указано в таблице ниже. Используйте любой возможный уровень декомпозиции – выберите тот, который окажется наиболее точным среди них.

1	2	3	4	5	6	7	8
<i>db8</i>	<i>coif3</i>	<i>haar</i>	<i>sym4</i>	<i>coif1</i>	<i>db4</i>	<i>db2</i>	<i>sym5</i>
9	10	11	12	13	14	15	16
<i>coif5</i>	<i>haar</i>	<i>sym6</i>	<i>db6</i>	<i>sym7</i>	<i>db10</i>	<i>sym8</i>	<i>db5</i>
17	18	19	20	21	22	23	24
<i>coif4</i>	<i>sym10</i>	<i>db16</i>	<i>haar</i>	<i>db20</i>	<i>sym16</i>	<i>coif2</i>	<i>db18</i>

- 13) Самый мощный инструмент вейвлет-декомпозиции, называемый **Пакетной Вейвлет Декомпозицией** (Wavelet Packet Decomposition = **WPD**). Этот метод опирается на дискретное преобразование, но позволяет произвести полный перебор комбинаций коэффициентов аппроксимации **a** и деталей **d**. Например, для 4 уровня это будет 16 комбинаций вида: 'aaaa', 'aaad', 'aada', 'aadd', 'adaa', 'adad', 'adda', 'addd', 'daaa', 'daad', 'dada', 'dadd', 'ddaa', 'ddad', 'ddda', 'dddd'

Для сравнения, на 4 уровне обычной дискретной вейвлет декомпозиции из пункта 7 доступна только одна комбинация 'addd'.

- 14) При таком большом числе возможных вариантов, декомпозиция представляется в виде бинарного дерева, выборка или удаление узлов, из которого и будет менять после реконструкции полученные временные ряды. Создадим общую пакетную декомпозицию:

```
wp = pywt.WaveletPacket(data=F, wavelet='dmey', mode='periodization')  
print([node.path for node in wp.get_level(4, 'freq')]) # выводим все комбинации  
узлов, упорядоченные по их частотной ширине спектра
```

- 15) Попробуем удалить один из узлов и посмотреть, что получится:

```
del wp['aaaa'] # удалим самый «глубокий» узел
reF = wp.reconstruct() # и восстановим ряд ...
plt.figure(figsize = (10, 5))
plt.plot(t, F, 'k')
plt.plot(t, reF, 'b') # получим нечто периодическое, плохого качества
plt.show()
```

- 16) Если удаление узлов не приводит к желаемым результатам, возможно есть смысл делать отдельную выборку ветвей этих узлов:

```
wp = pywt.WaveletPacket(data=F, wavelet='dmey', mode='periodization')
new_wp = pywt.WaveletPacket(data=None, wavelet='dmey', mode='periodization')
new_wp['aaaa'] = wp['aaaa'].data # выбираем первую ветвь
new_wp.reconstruct(update=True) # обновляем данные
reF1 = new_wp.data # восстанавливаем под нее ряд 1
new_wp = pywt.WaveletPacket(data=None, wavelet='dmey', mode='periodization')
new_wp['aaad'] = wp['aaad'].data # выбираем вторую ветвь
new_wp.reconstruct(update=True) # обновляем данные
reF2 = new_wp.data # восстанавливаем под нее ряд 2
plt.figure(figsize = (10, 5))
plt.plot(t, F, 'k')
plt.plot(t, reF1, 'b') # компонента 1
plt.plot(t, reF2, 'r') # компонента 2
plt.show()
```

- 17) Прodelайте подобную декомпозицию WPD для своего варианта вейвлета с выбором/удалением нужных узлов. Уровень декомпозиции может отличаться. Выбор комбинаций коэффициентов аппроксимации и детализации тоже может отличаться.

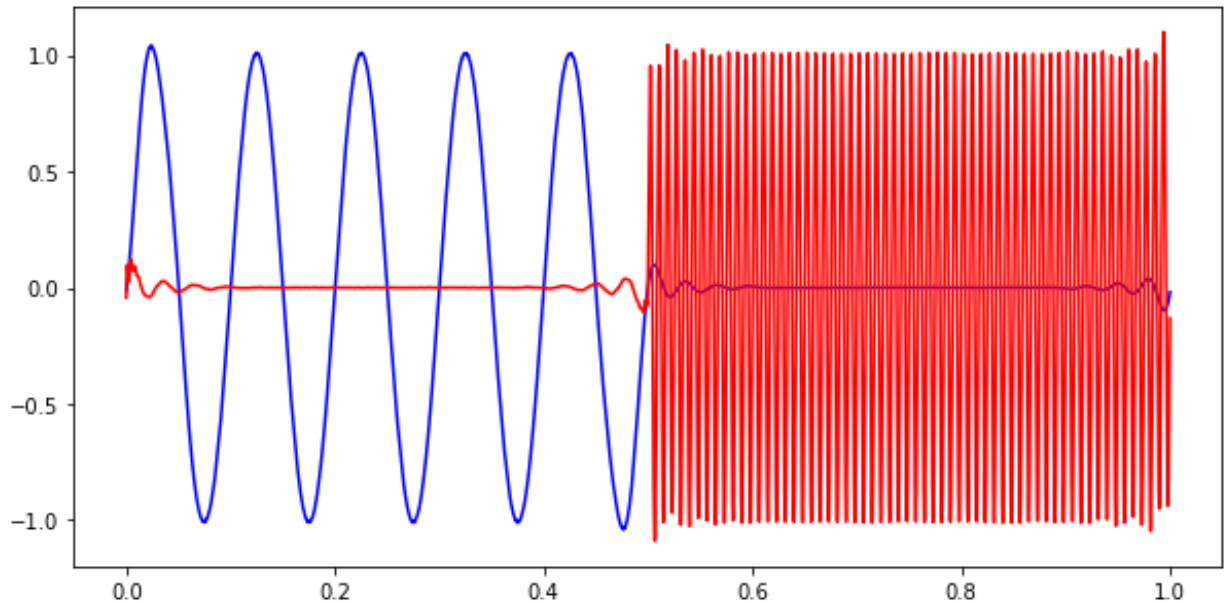
18) Теперь примените полученные навыки вейвлет-декомпозиции DWT и WPD, на основе базисного вейвлета для **своего варианта** из таблицы, для получения компонент временных рядов и пунктов ниже. Для каждого пункта повторите вейвлет-декомпозицию с другими параметрами, но с базисным вейвлетом Мейера (**dmey**). То есть для каждого сигнала/временного ряда в дальнейшем должны присутствовать как минимум 4 результата:

- Ваш вариант вейвлета, использующий функцию DWT (**wavedec**);
- Ваш вариант вейвлета, использующий функцию WPD (**WaveletPacket**);
- вейвлет Мейера, использующий функцию DWT (**wavedec**);
- вейвлет Мейера, использующий функцию WPD (**WaveletPacket**);

19) Декомпозируйте сигнал с частотным изломом на 2 периодические компоненты, разделенные по времени:

```
t = np.linspace(0, 1, 4096)
xf = np.zeros(4096)
for i in range(0, len(t)//2):
    xf[i] = np.sin(2*np.pi*10*t[i])
for i in range(len(t)//2, len(t)):
    xf[i] = np.sin(2*np.pi*120*t[i])
plt.figure(figsize = (10, 5))
plt.plot(t, xf)
plt.show()
```


Хороший результат выглядит примерно так:



- 20) Попробуйте выделить экспоненциальный тренд с помощью вейвлетов из следующего зашумленного временного ряда:

```
t = np.linspace(0, 4, 4096)
```

```
Fexp = np.exp(-0.4*np.pi*t) + 0.2*rand.randn(len(t))
```

Внимание! Для выделения тренда опцию **mode='periodization'** внутри функций вейвлетов **нужно будет убрать**, так как для тренда не нужен режим периодического дополнения точек по краям.

- 21) Смоделируйте временной ряд из **4 гармоник с шумом**, и разделите его на **4 гармоники** с помощью вейвлет декомпозиции:

$$u(t) = \sin[2\pi t(f_1)] + \sin[2\pi t(f_2)] + \sin[2\pi t(f_3)] + \sin[2\pi t(f_4)] + \xi(t)$$

```
t = np.linspace(0,1,1024)
f1 = 10
f2 = 40
f3 = 100
f4 = 150
F=2.0*np.sin(2*np.pi*f1*t)+1.5*np.sin(2*np.pi*f2*t)+0.8*np.sin(2*np.p
i*f3*t)+0.5*np.sin(2*np.pi*f4*t)+0.2*rand.randn(len(t))
plt.figure(figsize = (10, 15))
plt.subplot(5,1,1)
plt.plot(t, F, 'k')
plt.subplot(5,1,2)
plt.plot(t, 2.0*np.sin(2*np.pi*f1*t), 'b')
plt.subplot(5,1,3)
plt.plot(t, 1.5*np.sin(2*np.pi*f2*t), 'r')
plt.subplot(5,1,4)
plt.plot(t, 0.8*np.sin(2*np.pi*f3*t), 'g')
plt.subplot(5,1,5)
plt.plot(t, 0.5*np.sin(2*np.pi*f4*t), 'm')
plt.show()
```

Внимание! Здесь Вам потребуется выделять более двух компонент, из которых только одна будет определяться на основе коэффициента аппроксимации, а остальные – на основе коэффициентов детализации. Но коэффициенты детализации низких уровней имеют 2^n больше отсчетов (из-за квадратичной двоичной фильтрации), чем самый верхний уровень. Как тогда восстановить ряд? Тогда нужно просто использовать при восстановлении меньше **None**, чем необходимо.

Например, для 5 уровня можно построить более 6 компонент (аппроксимация + 5 деталей + комбинации деталей) следующим образом:

```
cA5, cD5, cD4, cD3, cD2, cD1 = pywt.wavedec(F, wvlt, mode='periodization', level=5)
pywt.waverec((cA5, None, None, None, None, None), wvlt, mode='periodization')
pywt.waverec((None, cD5, None, None, None, None), wvlt, mode='periodization')
pywt.waverec((None, cD4, None, None, None), wvlt, mode='periodization')
pywt.waverec((None, cD3, None, None), wvlt, mode='periodization')
pywt.waverec((None, cD2, None), wvlt, mode='periodization')
pywt.waverec((None, cD1), wvlt, mode='periodization')
```

или разные комбинации деталей:

```
pywt.waverec((None, cD4, cD3, None, None), wvlt, mode='periodization')
pywt.waverec((None, cD4, cD3, cD2, None), wvlt, mode='periodization')
pywt.waverec((None, cD3, None, cD1), wvlt, mode='periodization')
```

и т.д.

- 22) Создайте временной ряд как выборку нормального шума из 10000 точек:

```
ND = rand.randn(10000)
```

Разложите его на 8 отдельных компонент (1 аппроксимация + 7 деталей). Каждую компоненту постройте на фоне исходного временного ряда.

- 23) Постройте периодограмму и спектрограмму для каждой компоненты. Присутствуют ли какие-либо закономерности в получившихся графиках или между ними?

24) Загрузите временной ряд из файла doppler.mat:

```
file = h5py.File('doppler.mat','r')  
data = file.get('data')  
data = np.array(data)  
data = data.ravel()  
plt.figure(figsize = (10, 5))  
plt.plot(data, 'b')  
plt.show()
```

Декомпозируйте этот временной ряд с помощью вейвлетов таким образом, чтобы максимально **очистить его от шума**.

25) Загрузите временной ряд из файла Fort.mat:

```
file = h5py.File('Fort.mat','r')  
Z = file.get('Fort')  
Z = np.array(Z)  
Z = Z.ravel()  
plt.figure(figsize = (10, 5))  
plt.plot(Z, 'b')  
plt.show()
```

Декомпозируйте этот временной ряд с помощью вейвлетов на 3 компоненты: тренд + периодика + остаточные колебания.

26) Оформите итоговый отчет о проделанной работе.

2. Требования к оформлению отчета

Отчет в Jupyter-тетради должен обязательно содержать: номер лабораторной работы, ФИО студента, номер варианта (либо студенческий номер), номер группы, результаты выполнения работы с комментариями студента (комментарии пишутся после #) и изображениями.